СПЕЦКУРС З СУЧАСНИХ МЕТОДІВ І ТЕХНОЛОГІЙ РОЗРОБКИ ПРОГРАМНИХ ВИРОБІВ КОМПОНЕНТНОЇ АРХІТЕКТУРИ

О.П. Поліщук[α], С.О. Семеріков[β]
м. Кривий Ріг, Криворізький державний педагогічний університет
[α] apol@cabletv.dp.ua
[β] cc@kpi.dp.ua

Введення освітніх стандартів у педагогічних ВНЗ відіграло позитивну роль при плануванні навчання за моноспеціальностями “Фізика” та “Математика”. Проте на подвійних спеціальностях “Фізика та основи інформатики”, “Математика та основи інформатики” чіткі вимоги стандартів до кількості кредитів, з одного боку, та обмеження тижневого навантаження, з іншого, практично не залишили місця для додаткової спеціальності – інформатики.

При складанні та перегляді навчальних планів підготовки вчителів інформатики за подвійними спеціальностями (адже окремий напрямок підготовки “Інформатика” в педагогічних ВНЗ просто відсутній) відбувається перманентне скорочення ресурсів, відведених для навчання комп’ютерному програмуванню. У стандарті моноспеціальностей “Математика” та “Фізика” обсяг годин на інформатику менший, ніж на фізичне виховання, у навчальних планах подвійних спеціальностей – ненабагато більший, до того ж при їх складанні іде заміна програмістських дисциплін курсами загальноознайомчого характеру типу “Інформаційні технології і системи”, “Інформаційні технології в наукових дослідженнях”, (або в навчанні, або взагалі – “Сучасні інформаційні технології”) і т.п. Як правило, тематика таких дисциплін включає 15–18 розділів з досить серйозними назвами (“Інтелектуальні і експертні системи”, “Системи комп’ютерної графіки”, “Комп’ютерні мережі”, “Системи управління базами даних” і т.ін.). На всі такі розділи разом відводиться один семестр по одному практичному заняттю і від 0 до 1 лекції на тиждень – по одному заняттю на кожний розділ. Можна зустріти тематичні плани, в яких в курсах алгоритмізації і програмування практичні заняття відсутні зовсім. Така підміна приводить до підготовки в кращому випадку просто комп’ютерного користувача “побутового” рівня замість професійної підготовки фахівця в галузі високих технологій.

Будь-який технологічний процес виконується “в прямому напрямку” – від сировини до готового продукту, розробляється ж технологія “в зворотному напрямку” – від вимог до кінцевого виробу до розробки операцій і вимог до сировини і комплектуючих виробів. Технологія навчання не є виключенням. Тому ми спробуємо побудувати ескіз навчального плану для спеціальності “Інформатика”, що передбачає присвоєння випускнику кваліфікації “інженер-програміст, вчитель інформатики”, сформулювавши вимоги до його кваліфікації просто на основі “здорового глузду”.

Сучасні комп'ютеризовані системи, як правило, гетерогенні, тобто складаються як по апаратному, так і по програмному забезпеченню з різнорідних компонентів, виготовлених різними виробниками. Проблема об'єднання таких компонентів в єдине ціле до кінця не вирішена, але в цьому напрямку ведуться інтенсивні роботи і отримані результати широко використовуються на практиці з початку 90-х років. Будемо вважати, що інженер-програміст повинен в якійсь мірі бути знайомим з сучасними методологіями програмування і в 9-му семестрі має прослухати відповідний спецкурс. Ми підготували такий спецкурс для визначення мінімально необхідної підготовки студентів спеціальності "Інформатика" для його сприйняття.

Спецкурс під назвою "Методи розробки гнучких інтегрованих програмних виробів" присвячений вивченню компонентних технологій в розробці складних комп'ютерних програм, складові частини (компоненти) яких можуть бути реалізовані на різних мовах програмування, для різних платформ, модернізуватися незалежно одне від одного без необхідності внесення змін в систему, що використовує їхні сервіси.

Курс розрахований на 36 годин лекцій і 36 годин лабораторного практикуму з програмування. Курс лекцій включає 10 тем різної складності.

Вступна лекція присвячується огляду основних конкуруючих компонентних технологій розробки програмних виробів компонентної архітектури – (D)COM, CORBA, Java, висвітленню особливостей кожної з них, аналізу таких характеристик, як підтримка доступу клієнта до компонента через комп'ютерні мережі, мобільність, незалежність від мови програмування, платформи, трудомісткість освоєння програмістами. Обмежений об'єм відведених на предмет часових ресурсів не дозволяє детально розбиратися в теоретичних основах і особливостях використання кожної технології і змушує розглянути характерні риси компонентного підходу на одній з них. Такою вибрана технологія Microsoft під назвою (D)COM (Components object model – об'єктна модель компонентів), що дозволяє використовувати широкий спектр можливостей по розробці компонентів і їх взаємодії з клієнтом – від компонентів на рівні екземплярів класу до компонентів, розміщених в різних потоках процесу, або в різних процесах, що виконуються на одному або на різних комп'ютерах.

Курс побудовано по принципу поступового ускладнення матеріалу і в першому змістовному розділі розглядається важливість уніфікації інтерфейсів між компонентами і клієнтами і найпростіший метод реалізації інтерфейсів (клієнта і компонентів в одному процесі і в одному потоці) на основі абстрактних базових класів з множинним наслідуванням, без використання динамічної компоновки. Розглядаються позитивні риси технології інкапсуляції інтерфейсів (можливість вдосконалювати їх функціональність без порушення роботи системи) і недоліки (можливість взаємодії клієнта і компонента в обхід інтерфейсів).

Наступний розділ присвячений удосконаленню цієї взаємодії через спеціальний інтерфейс (доступ до якого отримується при створенні компонента), що в свою чергу використовується для запиту про підтримку потрібного клієнту інтерфейсу. Приділяється увага важливому питанню обліку посилань на інтерфейси компонентів і визначенню часу життєдіяльності компонентів. Тільки після цього здійснюється перехід до розміщення компонентів в бібліотеках динамічної компоновки і розглядається процес експорту функцій і взаємодії кількох клієнтів і компонентів в різних комбінаціях. Розміщення компонентів в динамічних бібліотеках дає можливість заміняти їх під час виконання, забезпечуючи велику гнучкість інтегрованої програмної системи.

Після приділення необхідної уваги деяким деталям (наприклад методам генерації неповторюваних кодів інтерфейсів, розгляду структури системного реєстру, реєстрації компонентів-серверів в реєстрі) здійснюється перехід до наступного ускладнення – використанню фабрики класів (компонента, який служить для створення інших компонентів) і її стандартних інтерфейсів.

Наступний етап – вивчення методів включення і агрегування як інструментів ієрархічної інтеграції для повторного використання компонентів, які самі можуть бути клієнтами компонентів.

Після цього ідуть розділи, присвячені досить питанням розміщення компонентів-серверів в різних процесах, локальним і віддаленим викликам процедур, техніці створення замісників і заглушок, основам мови опису інтерфейсів IDL (Interface Definition Language) і використанню компілятора MIDL для автоматичної генерації динамічних бібліотек “замісника/заглушки”, перетворенню через реєстр локальних серверів у віддалені, створенню диспетчерських інтерфейсів і автоматизації.

Заключна тема курсу – багатопоточність і розміщення компонентів в різних потоках, реалізація розділених і вільних потоків, підрозділи (конгломерат з потоку і циклу вибірки повідомлень) та питання потокобезпеки.

Всі лекції ілюструються по можливості фрагментами програм, приклади повної програмної реалізації ілюстрацій до кожної теми розміщуються на комп’ютерах робочих місць в класі для практичних занять по програмуванню.

Всі приклади носять “шкільний характер”, інтерфейси забезпечують просту функціональність (прийом чисел і рядків з клавіатури, сортування).

Методично-теоретичний матеріал базується переважно на роботах Дейла Роджерсона та Дональда Бокса. Приклади програм, що пропонуються Д. Роджерсоном, доповнені більш змістовною функціональністю інтерфейсів. В зв’язку з тим, що курс програмування в графічному середовищі навчальним планом не передбачений, графічний інтерфейс в прикладах замінено текстовим консольним (хоч це й не відповідає сучасним стандартам програмування).

Очевидно, що для сприйняття курсу студенті повинні мати базові знання і навички по основам об'єктно-орієнтованого програмування (вміти реалізовувати класи, розробляти конструктори, операторні функції, володіти методами одиночного і множинного наслідування класів в їх ієрархії тощо).

Об'єктно-орієнтоване програмування, як засіб подолання складності сучасних програмних виробів, в свою чергу вимагає надбання навичок роботи зі складними багатофайловими проектами, в яких присутня та складність, яку треба "долати". Базисом для переходу на об'єктну методологію повинна бути обізнаність з методами структурного процедурного програмування, фундаментальними алгоритмами і структурами даних, не кажучи вже просто про володіння технікою програмування на сучасних мовах високого рівня, знання бодай стандартних бібліотек функцій і класів, володіння допоміжними мовами типу IDL для опису інтерфейсів, синтаксисом створення файлів опису модулів, make-файлів тощо.

Необхідність розробки розподілених компонентів програмних систем вимагає освоєння багатозадачних операційних систем, теорії комп'ютерних мереж, основ динамічної компоновки і побудови динамічних бібліотек, принципів побудови компіляторів.

Звідси й випливає перелік і послідовність вивчення дисциплін, що складають цільову основу програмістської кваліфікації в чисто технічному плані – ми спеціально не зупинялись на основах теоретико-математичної та методичної підготовки.

Поза межами цієї статті залишилось багато невирішених питань організаційного і методичного характеру, питань технічного забезпечення процесу навчання комп'ютерному програмуванню, без вирішення яких ефективна підготовка потрібних країні фахівців з високих технологій неможлива.